This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

Additional data decoding for MPEG2 data stream of imag and sound information

Patent Number: DE19650515
Publication date: 1998-06-25
Inventor(s): BREUNINGER FRIEDEMANN (DE); FEHLHAMMER HEINZ DR (DE); KLEIN ROLF DIETER (DE); EITZ GERHARD (DE); BRUECKNER WERNER (DE); SCHAEFER RAINER DR (DE)
Applicant(s):: INST RUNDFUNKTECHNIK GMBH (DE)
Requested Patent: DE19650515
Application Number: DE19961050515 19961205
Priority Number (s): DE19961050515 19961205
IPC Classification: H04N7/025 ; H04N7/173
EC Classification: H04N5/00M8, H04N7/24T4, H04N7/52D
Equivalents:

Abstract

The method involves including additional data into a uniform data structure irrespective of its content. The service program for managing the collection memory (100) has a logical layer (121) with a language containing a limited number of logical commands, which is independent of the machine language for the processing unit, and which represents the only interface for access (1214) by other service programs to the additional data supplied. The logical layer has a self-managing search algorithm which accesses the content of the additional data without addressing and according to the structure of the additional data.

Data supplied from the **esp@cenet** database - I2

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENTAMT

(12) **Offenlegungsschrift**

(10) **DE 196 50 515 A 1**

(51) Int. Cl.[6]:
H 04 N 7/025
H 04 N 7/173

(21) Aktenzeichen: 196 50 515.1
(22) Anmeldetag: 5. 12. 96
(43) Offenlegungstag: 25. 6. 98

DE 196 50 515 A 1

(71) Anmelder:
Institut für Rundfunktechnik GmbH, 80939 München, DE

(74) Vertreter:
Konle, T., Dipl.-Ing., Pat.-Anw., 81247 München

(72) Erfinder:
Brückner, Werner, 81929 München, DE; Breuninger, Friedemann, 80801 München, DE; Eitz, Gerhard, 85586 Poing, DE; Fehlhammer, Heinz, Dr., 82041 Oberhaching, DE; Klein, Rolf Dieter, 80809 München, DE; Schäfer, Rainer, Dr., 85395 Wolfersdorf, DE

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Verfahren zum Decodieren von Zusatzdaten

(57) In digitalen Datenströmen aus Bild- und/oder Toninformationen, insbesondere bei den MPEG 2-Datenströmen sind Zusatz-Daten zusätzlich enthalten, welche beispielsweise neben Steuerinformationen auch ladbare, ablauffähige Rechenprogramme, Textinformationen und/oder Grafikinformationen umfassen. In einer empfängerseitigen Set-Top-Box werden
- die Zusatzdaten von dem Datenstrom abgetrennt,
- die abgetrennten Zusatz-Daten in dem Sammelspeicher einer Verarbeitungseinheit abgelegt;
- die abgelegten Zusatz-Daten nach Maßgabe von Dienstprogrammen von der Verarbeitungseinheit verwaltet und verarbeitet, wobei Dienstprogramme als residente Rechenprogramme und/oder als ladbare, ablauffähige Rechenprogramme vorliegen, und
- verarbeitete Zusatz-Daten zu ihrer Wiedergabe und/oder zur Steuerung der aus dem Datenstrom abgetrennten Bild- und/oder Toninformationen bereitgestellt.
Um die Decodierung der Zusatz-Daten für Set-Top-Box mit unterschiedlichem Mikro-Prozessor-Typ, unterschiedlichem Betriebssystem und unterschiedlicher Schnittstelle "API" gleichermaßen geeignet zu machen und eine Kompatibilität verschiedener Set-Top-Boxen zu erzielen, wird vorgeschlagen, daß die in einer einheitlichen Datenstruktur vorliegenden Zusatz-Daten unabhängig von ihrem jeweiligen Inhalt abgelegt werden. Das Dienstprogramm für die Verwaltung des Sammelspeichers stellt eine logische Schicht dar, welche von einer aus einer begrenzten Anzahl von logischen Befehlen ("FROM", "TO", "COPY") bestehenden ...

DE 196 50 515 A 1

Beschreibung

Die Erfindung bezieht sich auf ein Verfahren zum Decodieren von Zusatzdaten gemäß dem Oberbegriff des Patentanspruchs 1.

Beim digitalen Fernsehen werden Bild- und Toninformationen zusammen mit Zusatzdaten in einem Datenstrom entsprechend dem ISO/MPEG-2-Standard übertragen. Die empfangsseitige Verarbeitung dieses Datenstroms erfolgt in einem Aufsatzgerät (Set-Top-Box) zu einer herkömmlichen Fernsehempfangseinheit (TV-Empfänger, Videorecorder). Ein Blockschaltbild einer bekannten Set-Top-Box ist in **Fig.** 1 dargestellt. Der MPEG-2-Datenstrom **21**, welcher über einen Kabelanschluß, eine Satellitenantenne oder eine terrestrische Antenne mittels Tuner/Demodulator **10** empfangen und demoduliert wird, gelangt zu einem Demultiplexer **20**, welcher den Datenstrom **21** in die Bildinformationen **41**, die Toninformationen **42** sowie die Zusatzdaten **31** aufspaltet. Die Bildinformationen **41** werden in einen Bildspeicher **40** geladen, wo den Bildinformationen **41** Grafik- und Textinformationen **42** hinzugefügt werden. Der Inhalt des Bildspeichers **40** wird auf dem Bildschirm eines herkömmlichen Fernsehempfängers **50** wiedergegeben. Die abgetrennten Zusatzdaten **31** werden in einem Zwischenspeicher **30** gepuffert, um in einer Verarbeitungseinheit **100** verwaltet und verarbeitet zu werden. Die Zusatz-Daten **31** umfassen Steuerinformationen, ladbare, ablauffähige Rechenprogramme (Ablaufsteuerung **1**, Ablaufsteuerung **2**), Textinformationen (Text 1, Text 2 . . . Text k) sowie Grafikinformationen (Grafik 1, Grafik 2 . . . Grafik m). Die Text- und Grafikinformationen werden aus dem Zwischenspeicher **30** in einen RAM-Speicher **140** der Verarbeitungseinheit **100** vorübergehend geladen. Zusatzdaten **31**, welche dauernd in der Verarbeitungseinheit festgehalten werden sollen, insbesondere Zusatzdaten **31** von und für die Ablaufsteuerung, werden in einen EEPROM-Speicher **150** geladen. Die geladenen Zusatzdaten **31** werden in der Verarbeitungseinheit **100** behandelt, bevor sie dem Bildspeicher **40** zugeführt werden. Die ladbaren, ablauffähigen Rechenprogramme bewirken zusammen mit residenten Rechenprogrammen (welche in der Schnittstelle "API" zur Programmierung von Anwendungen enthalten sind) in Interaktion mit dem Benutzer das Auslesen, Aufbereiten und Wiedergeben der Text- und Grafikinformationen sowie die Steuerung der Wiedergabe von Bild- und/oder Toninformationen. Die Eingaben des Benutzers für die Interaktion, beispielsweise Fernbedienung und Uhr, werden in dem Betriebssystem der Verarbeitungseinheit **100** verwaltet. In der Schnittstelle "API" erfolgt ferner die Prüfung der Zugangsberechtigung des Benutzers für verschlüsselte TV-Programme sowie ggf. die Kommunikation über einen Rückkanal mit einem Systembetreiber. Die Schnittstelle "API" sowie das Betriebssystem der Verarbeitungseinheit **100** sind in einem ROM-Speicher nicht-flüchtig abgelegt.

Die Ablaufsteuerungen (nachladbare, ablauffähige Rechenprogramme) müssen bei der bekannten Set-Top-Box auf das Betriebssystem und die Schnittstelle "API" zugeschnitten sein, so daß eine Set-Top-Box nur die Zusatzdaten bestimmter Systembetreiber decodieren kann. Ferner muß der Binärcode der verwendeten Ablaufsteuerungen auf den in der Verarbeitungseinheit **100** für die Ausführung der Programme verwendeten Typ des Mikro-Prozessors **160** abgestimmt werden, so daß Änderungen des Mikro-Prozessor-Typs in Zukunft praktisch unmöglich sind oder für jeden Mikro-Prozessor-Typ eine eigene Ablaufsteuerung vom Sender nachgeladen werden muß. Des weiteren legt die jeweilige Schnittstelle "API" fest, wie die Grafik- und Textinformationen auf dem Bildschirm präsentiert werden. Der Systembetreiber muß daher hinsichtlich der Gestaltung der Grafik- und Textinformationen Rücksicht nehmen auf die Möglichkeiten der Schnittstelle "API", so daß spätere Änderungen bzw. Verbesserungen der Schnittstelle "API" aus Kompatibilitätsgründen mit bereits im Einsatz befindlichen Set-Top-Boxen stark eingeschränkt sind.

Die Aufgabe der Erfindung besteht demgegenüber darin, ein Verfahren der eingangs erwähnten Art zu schaffen, welches für Set-Top-Boxen mit unterschiedlichem Mikro-Prozessor-Typ, unterschiedlichem Betriebssystem und unterschiedlicher Schnittstelle "API" gleichermaßen geeignet ist, um eine Kompatibilität verschiedener Set-Top-Boxen zu erzielen.

Diese Aufgabe wird erfindungsgemäß durch die kennzeichnenden Merkmale des Patentanspruchs 1 gelöst.

Die Erfindung beruht auf der Grundüberlegung, in die Verarbeitungseinheit einer bekannten Set-Top-Box eine Software in Form einer logischen Schicht einzufügen, welche den Zugriff auf einen Sammelspeicher für die Zusatzdaten exklusiv verwaltet. Hierdurch werden die Zusatzdaten von der vorhandenen Soft- und Hardware der Set-Top-Box entkoppelt. Bei einer vorteilhaften Weiterbildung der Erfindung werden betreiberspezifische Steuerinformationen, welche in den Zusatzdaten enthalten sind, in Befehle für die logische Schicht sowie für weitere Dienstprogramme der Verarbeitungseinheit umgeformt. Diese Steuerinformationen gestatten dem Systembetreiber die Darstellung der Text- und Grafikinformationen sowie die Organisation von Auswahlhilfen für den Anwender weitestgehend frei zu gestalten.

Weitere Ausgestaltungen der Erfindung ergeben sich aus den Unteransprüchen 3 bis 10.

Die Erfindung wird nachstehend anhand eines in den Zeichnungen dargestellten Ausführungsbeispiels näher erläutert. Es zeigt:

Fig. 1 ein Blockdiagramm einer bekannten Set-Top-Box;

Fig. 2 ein Blockdiagramm eines Ausführungsbeispiels einer nach dem erfindungsgemäßen Verfahren arbeitenden Set-Top-Box;

Fig. 3a–3h Flußdiagramme für das erfindungsgemäße Verfahren, und

Fig. 4a–4d den schematischen Aufbau bestimmter Datenstrukturen ("Tuples"), welche bei dem erfindungsgemäßen Verfahren verwendet werden.

Der Aufbau und die Funktion der in **Fig.** 1 dargestellten bekannten Set-Top-Box ist bereits eingangs erläutert worden. Gegenüber diesem Stand der Technik weist die nach dem erfindungsgemäßen Verfahren arbeitende Set-Top-Box nach **Fig.** 2 einen völlig anderen Aufbau und Organisation der Verarbeitungseinheit **100** auf, wie nachfolgend im einzelnen erläutert werden soll. Die Hardware der Verarbeitungseinheit **100** umfaßt in gleicher Weise wie beim Stand der Technik (**Fig.** 1) zum Speichern die Einheiten ROM **130**, RAM **140**, EEPROM **150** und zum Ausführen einen Mikroprozessor **160**. Bestandteil der Speichereinheiten **130**, **140**, **150** sind ein Sammelspeicher **110** sowie die Gesamteinheit aller interner Dienstprogramme **120** der Verarbeitungseinheit **100**.

Die Zusatz-Daten **31** gelangen aus dem Zwischenspeicher **30** in einen Sammelspeicher **110**, welcher als sogenannter Tuple-Speicher organisiert ist. Physikalisch beseht der Sammelspeicher **110** aus nicht-flüchtigen und flüchtigen Speichereinheiten. Unter "Tuples" versteht man Datenstrukturen, welche bestehen aus

- einem Kopf, der ggf. in Unterköpfe unterteilt sein kann, und
- n Elemente (mit n = 1, 2 . . . k) unterschiedlichen Typs, beispielsweise

a) Typ "long" = 32 Bit lange Zahl mit Vorzeichen;
b) Typ "string" = lückenlose Aneinanderkettung von Zeichen (Buchstaben, Zahlen etc.), welche in einem Zeichensatz nach ASCII, ANSI oder einem anderen Standard definiert sind, wobei die Aneinanderkettung beliebige Länge aufweisen kann;
c) Typ "binary" = beliebiger Byte-Code mit beliebiger Länge, dem ein Einsprung-Header für die Ausführung vorangestellt ist; der Byte-Code umfaßt eine Relozierungs-Tabelle, die zur Umformung von absoluten Sprungadressen in relative Sprungadressen erforderlich ist. (Der Unterschied zwischen absoluten und relativen Sprungadressen wird später erläutert). Byte-Codes können beispielsweise sein:

- direkt vom Mikroprozessor der Verarbeitungseinheit **100** ausführbares Rechenprogramm in Maschinensprache,
- Helligkeits- und ggf. Farbwerte von Bildpunkten sowie deren Koordinatenwerte zum Aufbau einer Grafik,
- Abtastwerte eines digitalisierten Tonsignals zum Signalisieren von Audiohinweisen für den Benutzer,
- Informationen für Fernwirkvorgänge, z. B. Aktivierung eines Rückkanals.

Der grundsätzliche Aufbau von "Tuples" ist in den **Fig.** 4a bis 4d schematisch dargestellt. **Fig.** 4a zeigt den vorstehend bereits erwähnten Aufbau eines Tuples mit Kopf A/B, unterteilt in die Unterköpfe A (private header) und B (public header), sowie den Tuple-Elementen B (public header) und C (data). "Data" bedeuten bestimmte Inhaltsinformationen im Gegensatz zu Verwaltungsinformationen, welche in den Köpfen A und B enthalten sind. Inhaltsinformationen sind z. B. ausführbare Rechenprogramme oder Grafikinformationen. Verwaltungsinformationen sind für die Verwaltung der später erläuterten logischen Schicht **121** vorgesehen. Die Zuordnung des "public header" B sowohl zum Kopf A/B als auch zu den Elementen erklärt sich gemäß **Fig.** 4a damit, daß der "public header" B aus fünf fest vorgesehenen Tuple-Elementen "Lageflag", "Verfallsdatum", "PID" = "process-identifier", "APID" = "application identifier" und "Reserved" = Daten-Reserve für künftige Anforderungen besteht. Der "private header" A verwaltet den gesamten Tuple, wohingegen der "public header" B die einzelnen Tuple-Elemente verwaltet und daher auch eine "Kopf"-Funktion hat. Der "private header" A umfaßt gemäß **Fig.** 4b folgende Angaben zur Verwaltung des gesamten Tuple:

- Länge des Tuples, im Falle von **Fig.** 4a die Gesamtlängen von A+B+C;
- Anzahl n der Tuple-Elemente in B und C
- eine Anzahl von n Triple-Angaben für jedes Tuple-Element aus den n Tuple-Elementen, wobei jede Triple-Angabe besteht aus
  - der Länge des zugehörigen Tuple-Elementes
  - Typ des zugehörigen Tuple-Elementes
  - Match-Flag des zugehörigen Tuple-Elementes, wobei unter "Match-Flag" eine gesetzte Information verstanden wird, welche anzeigt, ob das zugehörige Tuple-Element (i) abgefragt, (ii) nicht abgefragt, oder (iii) überschrieben werden kann.

Auf die fünf fest vorgegebenen Tuple-Elemente des "public header" B folgen ab dem siebten Tuple-Element die Inhaltsinformationen ("data"), wie sich aus **Fig.** 4d ergibt. Das sechste Tuple-Element (**Fig.** 4d) beinhaltet eine symbolische Kurzangabe zur Klassifizierung der nachfolgenden Inhaltsinformationen (data), z. B.

- HTML = hypertext markup language (abgeleitete Seitenbeschreibungssprache), welche in der später erläuterten Grafikoberfläche **123** (**Fig.** 2) verwendet wird;
- BMP = Bitmap (Grafik);
- LZ = Lesezeichen;
- UMFORM = Identifikation der USBL (universal set-top-box-language), welche in dem später erläuterten Umformer **124** (**Fig.** 2) verwendet wird.

Die Verarbeitungseinheit **100**, welche den Sammelspeicher **100** umfaßt, weist als wesentlichen Unterschied zum Stand der Technik eine logische Schicht **121** auf, welche ausschließlich den Sammelspeicher **110** verwaltet.

Die logische Schicht **121** umfaßt eine Anzahl unabhängiger Routinen, nämlich

- einen Initialisierer **1211**,
- eine Empfangsroutine **1212**,
- eine Freispeicherverwaltung **1213**, und
- eine Zugriffsroutine.

Der logische Ablauf innerhalb der vorgenannten Routinen **1211**, **1212**, **1213** und **1214** ist an Hand von Flußdiagrammen gemäß **Fig.** 3a (Initialisierer), **Fig.** 3e (Empfangsroutine), **Fig.** 3f (Freispeicherverwaltung) und **Fig.** 3h (Zugriffsroutine) veranschaulicht. Die Einzelheiten des logischen Ablaufs der Routinen **1211** bis **1214** sollen später erläutert werden.

Die logische Schicht **121** kommuniziert innerhalb der Verarbeitungseinheit **100** mit einer Grafikoberfläche **123** und einem Umformer **124**, welche interne Dienstprogramme darstellen und deren logischer Ablauf wiederum an Hand von Flußdiagrammen gemäß **Fig.** 3b (Grafikoberfläche **123**) und **Fig.** 3c (Umformer) veranschaulicht ist.

Die Verarbeitungseinheit **100** weist schließlich als weitere interne Dienstprogramme eine Systemsteuerung **122** und ein Betriebssystem **125** auf, deren logischer Ablauf an Hand von Flußdiagrammen gemäß **Fig.** 3d (Systemsteuerung) und **Fig.** 3g (Betriebssystem) erläutert wird.

Der physikalische Zugriff der Verarbeitungseinheit **100** auf die verschiedenen Speichereinheiten **130** (ROM), **140** (RAM) und **150** (EEPROM) erfolgt einerseits durch das Betriebssystem **125** und andererseits durch den Mikroprozessor **160**. Der logische Zugriff auf den Sammelspeicher **110** erfolgt durch die Zugriffsroutine **1214**, wie noch erläutert werden soll.

Der logische Ablauf der Routinen **1211** bis **1214** soll nunmehr an Hand der Flußdiagramme nach **Fig.** 3a, 3e, 3f und 3h näher erläutert werden.

Das Einschalten der Set-Top-Box löst zuerst einen Rücksetzvorgang-Vorgang "Hardware-Reset") aus (**Fig.** 3a), worauf das Betriebssystem **125** initialisiert wird (Routinenaufruf "Init-Betriebssystem"). In weiterer Abfolge werden die vier Routinen **1211** bis **1214** der logischen Schicht **121** initialisiert (Aufrufe "Init-Freispeicherverwaltung", "Init_Zugriffsroutine" "Init_Empfangsroutine" und "Init_Systemsteuerung"), worauf die internen Dienstprogramme **122**, **123**, **124** gestartet werden (Aufrufe "Enter_Task (Systemsteuerung)", "Enter_Task (Grafikoberfläche)" und "Enter_Task (Umformer)"). Anschließend wird die Prozedur "Schedule Task" aufgerufen, welche die Mehrfachaufgabenverteilung (cooperative multitasking) der Gesamtheit aller internen Dienstprogramme **120** steuert.

Nach erfolgter Initialisierung laufen die internen Dienstprogramme Systemsteuerung **122**, Grafikoberfläche **123** und Umformer **124** und warten auf die Zuweisung von Aufgaben, wie später noch erläutert wird.

Die Empfangsroutine **1212** hat die Aufgabe, den Inhalt des Zwischenspeichers **30** zu lesen und in geeigneter Form im Sammelspeicher **110** abzulegen. Hierzu wird vom Zwischenspeicher **30** der Hardware-Befehl "Interrupt" (**Fig.** 3e) ausgelöst, welcher den Mikroprozessor **160** veranlaßt, in einer Routine "Baue Tuple zusammen" die Datenstrukturen (Tuples) gemäß **Fig.** 4a bis 4d aus den Zusatzdaten **31** zu bilden. Die Empfangsroutine **1212** entscheidet in einer nachfolgenden Prozedur, ob die Tuple-Bildung abgeschlossen ist. Falls die Entscheidung "nein" lautet, kehrt die Empfangsroutine **1212** zu der Prozedur "Schedule-Task" (**Fig.** 3a) zurück und wartet auf einen neuen "Interrupt"-Befehl. Falls die Entscheidung "ja" lautet, wird in einer Routine "Erzeuge private header" ein Unterkopf A (private header) gemäß **Fig.** 4a und 4b für den Tuple erzeugt. Darauf werden die Tuple im Sammelspeicher **110** abgelegt. Ferner prüft die Empfangsroutine **1212**, ob in dem Sammelspeicher **110** ein gleiches Tuple bereits existiert und bewirkt über eine Prozedur in der Freispeicherverwaltung **1213** die Löschung dieses identischen Tuples im Sammelspeicher **110**. Anschließend kehrt die Empfangsroutine zu der Prozedur "Schedule_Task" (**Fig.** 3a) zurück und wartet auf den nächsten "Interrupt"-Befehl.

Die Freispeicherverwaltung **1213** erhält von den aufrufenden Dienstprogrammen **120** die Anweisung ("Collect Call (size)"), einen bestimmten Umfang (size) an Speicherplätzen im Sammelspeicher **110** frei zu machen. Dies erfolgt zunächst durch geeignetes Verschieben von freigegebenen Speicherplätzen (Defragmentieren). Anschließend entscheidet die Freispeicherverwaltung **1213**, ob genügend (size) freier, zusammenhängender Speicherplatz vorhanden ist (Entscheidungsprozedur "Ist Platz frei?"). Falls "ja" kehrt die Freispeicherverwaltung **1213** zum aufrufenden Dienstprogramm zurück. Falls "nein", werden zunächst veraltete Tuples (mit abgelaufenem Verfallsdatum) gelöscht. In einer zweiten Entscheidungsprozedur "Ist Platz frei?" wird bei Bejahung zum aufrufenden Dienstprogramm zurückgekehrt, bei Verneinung werden die ältesten Tuples und Lageflag "RAM" gelöscht. Nun folgt eine dritte Entscheidungsprozedur "Ist Platz frei". Bei Bejahung erfolgt Rückkehr zum aufrufenden Dienstprogramm; bei Verneinung wird zur letzten Prozedur zurückgesprungen und solange wiederholt, bis genügend (size) Speicherplatz im Sammelspeicher **110** geschaffen ist.

Die Zugriffsroutine **1214** hat die Aufgabe, einen Zugriff der Gesamtheit der internen Dienstprogramme **120** auf den Sammelspeicher **110** zu verwalten. Hierzu wird bei einer Zugriffsanforderung eines Dienstprogrammes der Anforderungsbefehl ("from_tuple", "to_tuple", "copy_tuple") gelesen. In drei kaskadierten Entscheidungsprozeduren wird je nach Inhalt des Anforderungsbefehls in die zugeordneten Prozeduren

"Lese Tuple aus Sammelspeicher und lösche ihn" ("from_tuple"),
"Schreibe Tuple in Sammelspeicher" ("to_tuple"), und
"Lese Tuple aus Sammelspeicher und belasse ihn dort" ("copy_tuple")

gesprungen. Danach kehren alle drei vorgenannten Prozeduren zu den aufrufenden Dienstprogrammen zurück.

Im folgenden sollen die Dienstprogramme Systemsteuerung **122**, Umformer **124** und Grafikoberfläche **123** beschrieben werden.

Die Funktion der Systemsteuerung **122** besteht darin (**Fig.** 3d), zunächst in der Entscheidungs-Prozedur "Init Aufruf?" zu unterscheiden, ob der Aufruf von "Init-Systemsteuerung" (**Fig.** 3a) erfolgt ist oder ein Aufruf von einem internen Dienstprogramm ausgeht. Im Falle eines von "Init Systemsteuerung ausgehenden Aufrufs wird der Inhalt von externen Kenngrößen, z. B. Uhr, Fernbedienung, Seriennummer (**Fig.** 2) von aufeinanderfolgenden Prozeduren "Lese Seriennummer", "Lese Fernbedienung" und "Lese Uhr" ausgelesen und zusammen mit einer symbolischen Kurzbeschreibung im Sammelspeicher **110** (**Fig.** 2) abgelegt. Daraufhin wird zur aufrufenden Prozedur "Init_Systemsteuerung" zurückgekehrt. Im Falle eines Aufrufs von einem internen Dienstprogramm, z. B. **123, 124** (**Fig.** 2) holt sich die Systemsteuerung **122** anhand einer symbolischen Kurzbeschreibung, z. B. "TUNER_SET" mittels des Befehls des "from_tuple" über die Zugriffsroutine **1214** für sich geeignete Tuple aus dem Sammelspeicher **110**. "Geeignet" meint alle Tuples, auf welche die symbolische Kurzbeschreibung zutrifft (vgl. **Fig.** 4d das Beispiel des 6. Tuple-Elementes "Symbolisches Kürzel zur Klassifizierung"). Findet die Systemsteuerung **122** ein geeignetes auf die symbolische Kurzbeschreibung zutreffendes Tuple, wird der Inhalt des gefundenen Tuple von einer Routine ausgeführt, z. B. "Tuner umschalten" (**Fig.** 3d). Dieser Vorgang des Suchens von geeigneten Tuple im Sammelspeicher **110** und des Auslesens des Inhalts von gefundenen Tuple wird solange wiederholt, bis keine geeigneten Tuple mehr gefunden und ausgeführt werden können. Dann wird in die Routine "Schedule Task" (**Fig.** 3a) verzweigt, in welcher auf das Einlesen von neuen Tuple in den Sammelspeicher **110** gewartet wird. Werden nun neue Tuple eingelesen, startet die Systemsteuerung **122** – wie beschrieben – selbständig und überprüft erneut das Vorhandensein für sie geeigneter Tuple.

Die Funktion des Umformers **124** besteht darin (**Fig.** 3c), die im Zusammenhang mit der Klassifizierung eines Tuple (**Fig.** 4d) bereits erwähnte USBL (= Universal Set-top-Box Language) zu interpretieren und auszuführen. Die USBL stellt einen Satz von Befehlen auf hochabstrahierendem Niveau dar, wie es z. B. die Programmiersprachen C oder C++ oder JAVA sind. Bei USBL handelt es sich um eine sinnvoll reduzierte Teilmenge der Programmiersprache C. In USBL wird vom Programmanbieter insbesondere das Programm für die Verarbeitung von elektronischen Lesezeichen gesendet, welche dem Benutzer eine thematische Auswahl aus der Fülle empfangbare Fernsehprogramme auf elektronische Weise ermöglicht. Des weiteren kann in USBL ein internes Dienstprogramm vom Programmanbieter aus erneuert bzw. erweitert werden.

Zum Interpretieren und Ausführen der USBL-Befehle holt sich der Umformer **124** anhand einer symbolischen Kurzbeschreibung, z. B. "LZ" (vgl. **Fig.** 4d), mittels des Befehls "from_tuple" über die Zugriffsroutine **1214** für sie geeignete Tuple aus dem Sammelspeicher **110**. Findet der Umformer **124** ein geeignetes, auf die symbolische Kurzbeschreibung zutreffendes Tuple, wird der Inhalt des gefundenen Tuple in einer Routine "USBL interpretieren und ausführen" interpretiert. Dies bedeutet, daß der hochabstrahierende Tuple-Inhalt in USBL von einer Parser-Routine semantisch zerlegt wird in ausführbare Anweisungen niedrigerer Abstraktion, die in HMTL (vgl. **Fig.** 4d) im Sammelspeicher **110** abgelegt werden. HMTL ist eine abgeleitete Seitenbeschreibungssprache und keine algorithmische Sprache, wie sie USBL darstellt. Das Ablegen der ausführbaren Anweisungen niedriger Abstraktion erfolgt deshalb in HMTL, damit das interne Dienstprogramm **123** (Grafikoberfläche) unabhängig von seiner jeweiligen programmiersprachlichen Implementierung auf die vom Umformer **124**

für das interne Dienstprogramm **123** im Sammelspeicher **110** abgelegten Anweisungen zugreifen kann. Die Implementierungsunabhängigkeit der Grafikoberfläche **123** ermöglicht es, unterschiedliche Werkzeuge zum Erzeugen von grafischen Benutzeroberflächen mit gleichen logischen Funktionsabläufen (z. B. die Werkzeuge "OpenTV" oder "MediaHighway") in der Set-Top-Box ohne Änderung der übrigen internen Dienstprogramme **121**, **122**, **124**, **125** zu verwenden.

Auf die vorstehend erwähnte Routine "USBL interpretieren und ausführen" (**Fig.** 3c) folgt eine Entscheidungsprozedur "Weiterer Befehl?", ob der Tuple-Inhalt in USBL einen weiteren Befehl enthält. Falls "ja", wird an den Anfang der Routine "USBL interpretieren und ausführen" gesprungen und die Routine solange wiederholt, bis alle in USBL enthaltenen Befehle des aktuell bearbeiteten Tuple interpretiert und ausgeführt sind. Falls "nein", holt sich der Umformer **124** einen neuen Tuple aus dem Sammelspeicher **110** mit dem symbolischen Kürzel "LZ". Dieser Vorgang wird wiederum solange wiederholt bis kein geeignetes Tuple mit dem symbolischen Kürzel "LZ" im Sammelspeicher **110** mehr gefunden wird. Dann wird in die Routine "Schedule_Task" (**Fig.** 3a) verzweigt, in welcher auf das Einlesen von neuen Tuple in den Sammelspeicher **110** gewartet wird.

Die Funktion der Grafikoberfläche **123** besteht darin (**Fig.** 3b), vom Umformer **124** in der Sprache HTML im Sammelspeicher **110** abgelegte Grafikanweisungen auszuführen. Beispielsweise besteht diese Ausführung darin, eine Auswahlliste verschiedener Fernsehprogramme, welche thematisch einem bestimmten Lesezeichen zugeordnet sind, zur Interaktion mit dem Benutzer (z. B. Anklicken eines Bestätigungsfeldes auf dem Bildschirm mit der Fernsteuerung) darzustellen. Dazu holt sich die Grafikoberfläche **123** in einer Routine "from tuple (. . ."HTML". . .)" aus dem Sammelspeicher **110** anhand einer symbolischen Kurzbeschreibung "HTML" mittels des Befehls "from_tuple" für sich geeignete Tuple aus dem Sammelspeicher **110**. Findet die Grafikoberfläche **123** in der Entscheidungsroutine "gefunden?" ein geeignetes, auf die symbolische Kurzbeschreibung "HTML" zutreffendes Tuple, wird in einer Routine "Ausführung" der Inhalt des gefundenen Tuple, nämlich die Beschreibung eines oder mehrerer Grafikelemente, umgesetzt aus der abstrakten Beschreibung in HTML in physikalische Bildschirmparameter, wie Bildschirmpositionen, Größenangaben, Zeichengrößen und Funktionalität (z. B. radio button, check box, push button). Die Grafikelemente mit den errechneten physikalischen Bildschirmparametern werden mit Hilfe des Mikroprozessors **160** zu einem bestimmten Zeitpunkt mit den Bildsignalen **41** im Bildspeicher verknüpft. Gegebenenfalls wird in einer Routine "to_tuple (Antwort)" nur eine Antwort der Grafikoberfläche **123** – ausgelöst durch eine Reaktion des Benutzers – im Sammelspeicher **110** als Anweisung für andere interne Dienstprogramme, z. B. die Systemsteuerung **122**, abgelegt. Findet die Grafikoberfläche **123** in der Entscheidungsroutine "gefunden?" kein auf die symbolische Kurzbezeichnung "HTML" zutreffendes Tuple, holt sich die Grafikoberfläche **123** in einer Routine "from_tuple. . .BMP" über die Zugriffsroutine **1214** mittels des Befehls "from_tuple" Tuple mit der symbolischen Kurzbeschreibung "BMP" (=Bitmap). Findet die Grafikoberfläche **123** in der Entscheidungsroutine "gefunden?" ein geeignetes, auf die symbolische Kurzbeschreibung "BMP" zutreffendes Tuple, wird in einer Routine "Anzeigen" der Inhalt des gefundenen Tuple, nämlich eine Bitmap, in Form eines Grafik-Elementes aus einer bestimmten Bildschirmposition und zu einem bestimmten Zeitpunkt mit den Bildsignalen **41** im Bildspeicher **40** verknüpft. Findet die Grafikoberfläche **123** kein geeignetes, auf "BMP" zutreffendes Tuple, so erfolgt in ähnlichen Routinen wie für "HTML" und "BMP" eine Abfrage nach Tuple mit anderen symbolischen Kurzbeschreibungen, z. B. für komprimierte Bilder. Wird überhaupt kein geeignetes Tuple gefunden, verweist die Grafikoberfläche **123** in die Routine "Schedule_Task" (**Fig.** 3e), in welcher auf das Einlesen von neuen Tuple in den Sammelspeicher **110** gewartet wird.

Das Betriebssystem **125** hat nach erfolgter Initialisierung die Aufgabe, die Anweisungen von der Systemsteuerung mittels Routinen **122** auf die Hardware-Komponenten der Set-Top-Box, wie z. B. Tuner **10** oder Demultiplexer **20** umzusetzen (**Fig.** 3g).

Bei der Anwendung des erfindungsgemäßen Verfahrens auf bekannte Set-Top-Boxen können dort alle vorhandenen Hard- und Software-Komponenten weiterverwendet werden. Es wird lediglich eine logische Schicht **121** mit einer Anzahl unabhängiger Routinen **1211–1214** eingefügt, welche das Verbindungsglied zwischen internen Dienstprogrammen und einem Sammelspeicher für Tuple bildet. Dadurch werden die internen Dienstprogramme voneinander vollständig entkoppelt, was wiederum Voraussetzung dafür ist, den Umformer **124** einzufügen. Der Umformer **124** ist ein weiteres Entkopplungsglied zum Programmanbieten, weil der Anbieter durch die Hochsprache USBL freie Gestaltungsmöglichkeiten für die Darstellung von Text- und Grafikinformationen sowie die Organisation von Auswahlhilfen ("TV-Guides", "Navigatoren") hat. Durch die Kommunikation zwischen den internen Dienstprogrammen mit dem Sammelspeicher ausschließlich über die logische Schicht **121** läßt sich eine einfache Mehrfachaufgabenverteilung ("cooperative multitasking") ermöglichen. Durch die Organisation der Zusatzdaten **31** im Sammelspeicher **110** als Tuple läßt sich ein schneller Zugriff mit eigener Intelligenz ("Suchen beim Zugreifen" = "matching") auf die gespeicherten Zusatzdaten erzielen. Des weiteren ist es durch die Tuple-Organisation möglich, eine dynamische Datenbank zu verwalten, die nicht wie im herkömmlichen Sinne eine fest vorgegebene Anzahl von statischen Feldern aufweist, sondern eine beliebige Anzahl von Feldern als Datensatzstruktur (Tuple) besitzt. Die Intelligenz des erfindungsgemäß vorgesehenen Zugriffs auf die Zusatzdaten **31** ermöglicht ferner ein selbständiges Löschen veralteter Zusatzdaten **31**, da Tuple eine Zeitinformation (Verfallsdatum) in sich tragen.

Patentansprüche

1. Verfahren zum Decodieren von Zusatzdaten, welche in einem Datenstrom aus Bild- und/oder Toninformationen, insbesondere MPEG-2-Datenstrom, zusätzlich enthalten sind und welche beispielsweise neben Steuerinformationen auch ladbare, ablauffähige Rechenprogramme, Textinformationen und/oder Grafikinformationen umfassen, bei dem

- die Zusatzdaten von dem Datenstrom abgetrennt werden (Stufe **20**);
- die abgetrennten Zusatzdaten in dem Sammelspeicher (**110**) einer Verarbeitungseinheit (**100**) abgelegt werden;
- die abgelegten Zusatzdaten nach Maßgabe von Dienstprogrammen von der Verarbeitungseinheit (**100**) verwaltet und verarbeitet werden, wobei Dienstprogramme als residente Rechenprogramme und/oder als ladbare, ablauffähige Rechenprogramme vorliegen, und
- verarbeitete Zusatzdaten zu ihrer Wiedergabe und/oder zur Steuerung der aus dem Datenstrom abgetrennten Bild- und/oder Toninformationen

bereitgestellt werden (Stufe **40**).

dadurch gekennzeichnet, daß die in einer einheitlichen Datenstruktur vorliegenden Zusatzdaten unabhängig von ihrem jeweiligen Inhalt abgelegt werden, daß das Dienstprogramm für die Verwaltung des Sammelspeichers (**100**) eine logische Schicht (**121**) darstellt, welche von einer, aus einer begrenzten Anzahl von logischen Befehlen ("FROM", "TO", "COPY") bestehenden ersten Sprache angesprochen wird, die von der Maschinensprache der Verarbeitungseinheit (**100**) unabhängig ist und welche die alleinige Schnittstelle darstellt für den Zugriff (**1214**) der übrigen Dienstprogramme auf die abgelegten Zusatzdaten, und daß die logische Schicht (**121**) einen selbstverwaltenden Suchalgorithmus (Sammelspeicher-Zugriff **1214**) enthält, welcher adreßfrei nach Maßgabe der Struktur der abgelegten Zusatzdaten auf deren Inhalt zugreift.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß als weiteres Dienstprogramm ein Umformer (**124**) vorgesehen ist, welcher mit der logischen Schicht (**121**) kommuniziert und Steuerinformationen, welche in abgelegten Zusatzdaten enthalten sind, aus dem Sammelspeicher (**110**) liest und zu Befehlen für Dienstprogramme der Verarbeitungseinheit (**100**) umformt, wobei diese Befehle in dem Sammelspeicher (**110**) abgelegt werden.

3. Verfahren nach Anspruch 2, dadurch gekennzeichnet, daß die Befehle Bestandteil einer zweiten, aus einem begrenzten Befehlssatz bestehenden Sprache (HTML) sind.

4. Verfahren nach Anspruch 2 oder 3, dadurch gekennzeichnet, daß als weiteres Dienstprogramm eine Grafikoberfläche (**123**) vorgesehen ist, welche mit der logischen Schicht (**121**) kommuniziert und nach Maßgabe der für die Grafikoberfläche (**123**) bestimmten, im Sammelspeicher (**110**) abgelegten Befehle des Umformers (**124**) Text- und/oder Grafikinformationen zur Wiedergabe aufbereitet.

5. Verfahren nach einem der Ansprüche 2 bis 4, dadurch gekennzeichnet, daß als weiteres Dienstprogramm eine Systemsteuerung (**122**) vorgesehen ist, welche den Betrieb der Verarbeitungseinheit (**100**) im Sinne einer Mehrfachaufgabenverteilung steuert, sowie mit der logischen Schicht (**121**) kommuniziert, um nach Maßgabe der für die Systemsteuerung (**122**) bestimmten, im Sammelspeicher (**110**) abgelegten Befehle des Umformers, (**124**) sowie gegebenenfalls in Abhängigkeit von externen Kenngrößen (beispielsweise aus einer Fernbedienung) weitere Befehle für Dienstprogramme, sowie Kommandos zur Steuerung externer Einheiten (beispielsweise Tuner) generiert, wobei die weiteren Befehle für Dienstprogramme in dem Sammelspeicher (**110**) abgelegt werden.

6. Verfahren nach einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, daß die logische Schicht (**121**) einen Initialisierer (**1211**) aufweist, welcher

- den Sammelspeicher (**110**) nach abgelegten Zusatzdaten durchsucht und vorhandene abgelegte Zusatzdaten zur Verwendung in der Verarbeitungseinheit (**100**) freigibt, und
- die freigegebenen Zusatzdaten nach ablauffähigen Rechenprogrammen durchsucht und gegebenenfalls startet.

7. Verfahren nach einem der Ansprüche 1 bis 6, dadurch gekennzeichnet, daß die logische Schicht (**121**) eine Freispeicherverwaltung (**1213**) aufweist, welche

- abgelegte Daten nach bestimmten Kriterien löscht, beispielsweise wenn ein Verfallsdatum erreicht ist oder nach Maßgabe einer Lösch-Strategie, und
- den Speicherraum des Sammelspeichers (**110**) für die abgelegten Daten derart reorganisiert, daß der zusammenhängende Speicherbereich eine maximale Größe aufweist.

8. Verfahren nach einem der Ansprüche 1 bis 7, dadurch gekennzeichnet, daß die logische Schicht (**121**) eine Empfangsroutine (**1212**) aufweist, welche nach Maßgabe eines von der Verarbeitungseinheit (**100**) unabhängigen Hardware-Befehls ("Interrupt") abgetrennte Daten ablegt und/oder erneuert und/oder ignoriert.

Hierzu 12 Seite(n) Zeichnungen

Num[redacted]
Int. Cl.:
Offenlegungstag:
DE 196 50 515 A1
H 04 N 7/025
25. Juni 1998

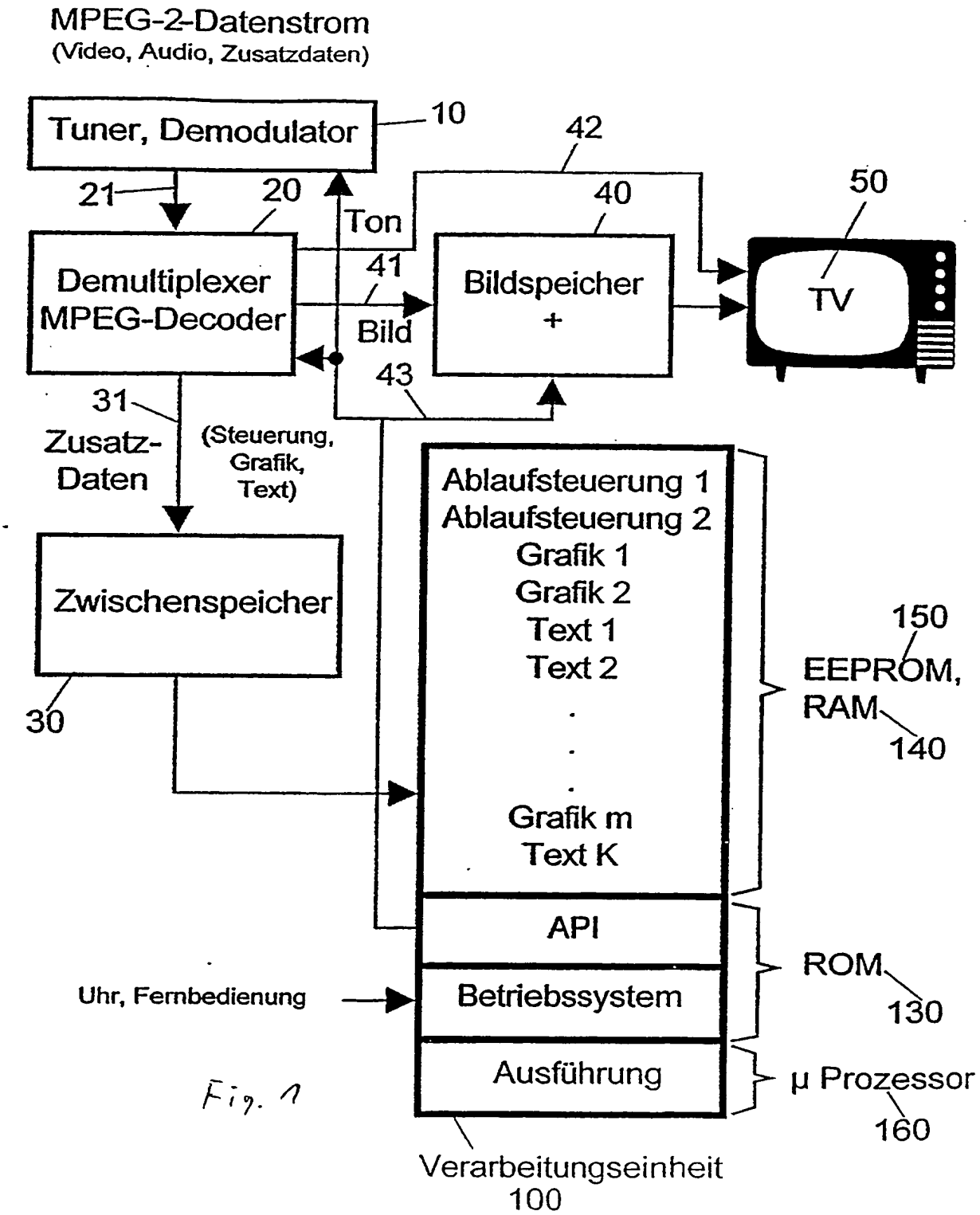

Fig. 1

Num[redacted]
Int. Cl.[redacted]
Offenlegungstag:
DE 196 50 515 A1
H 04 N 7/025
25. Juni 1998

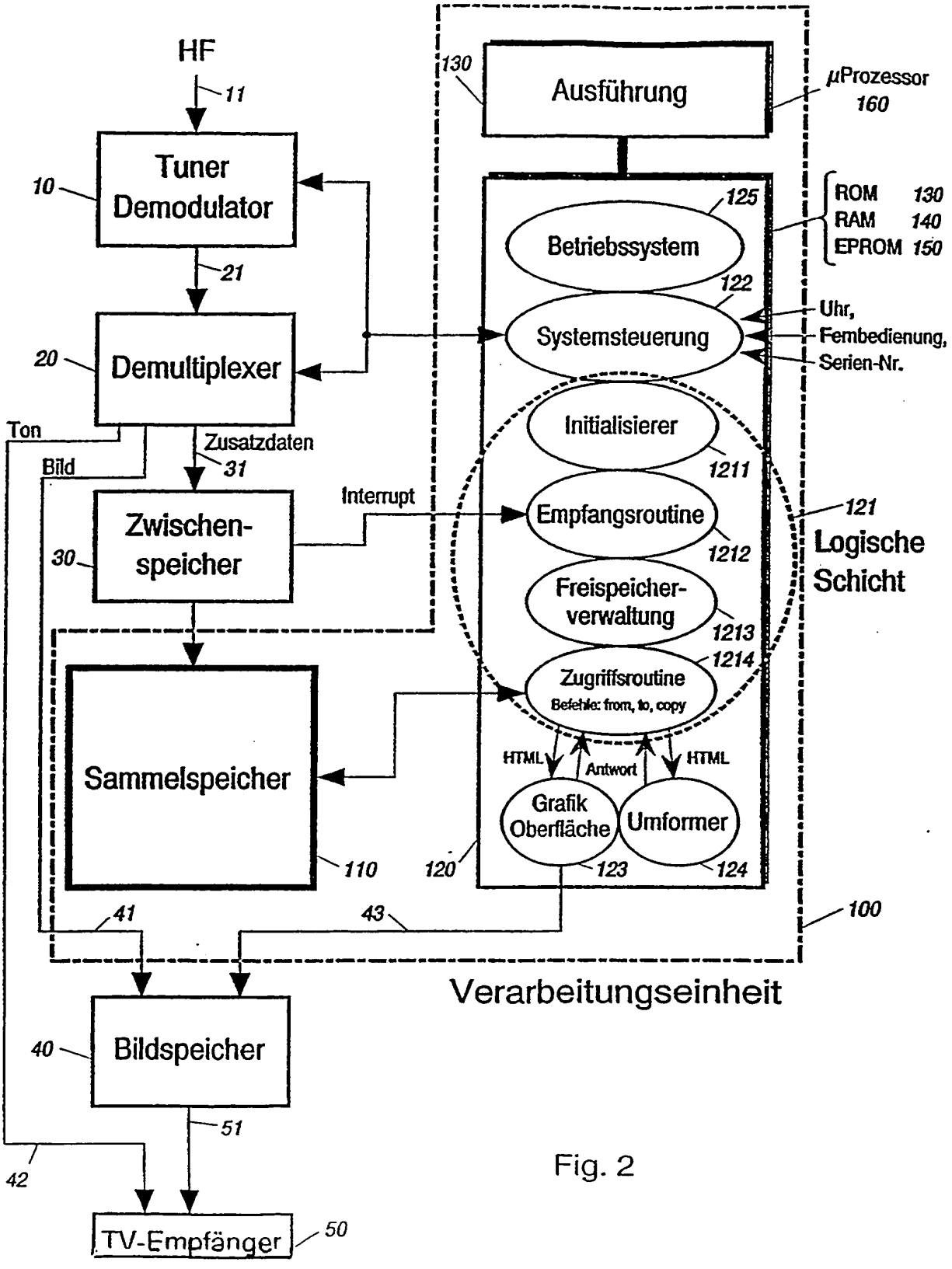

Fig. 2

Numm[redacted]
Int. Cl.:
Offenlegungstag:

DE 196 50 515 A1
H 04 N 7/025
25. Juni 1998

Initialisierer 1211

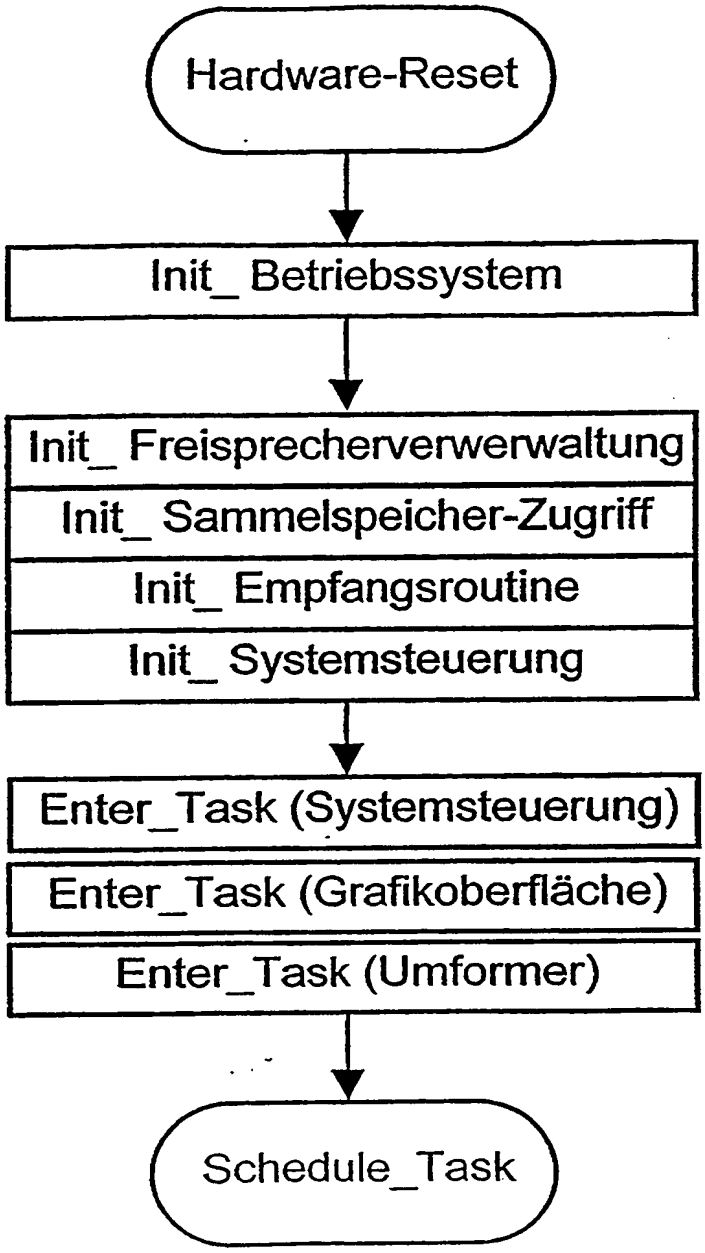

Fig. 3a

Num[redacted]
Int. Cl.
Offenlegungstag:
DE 196 50 515 A1
H 04 N 7/025
25. Juni 1998

Graphikoberfläche 123

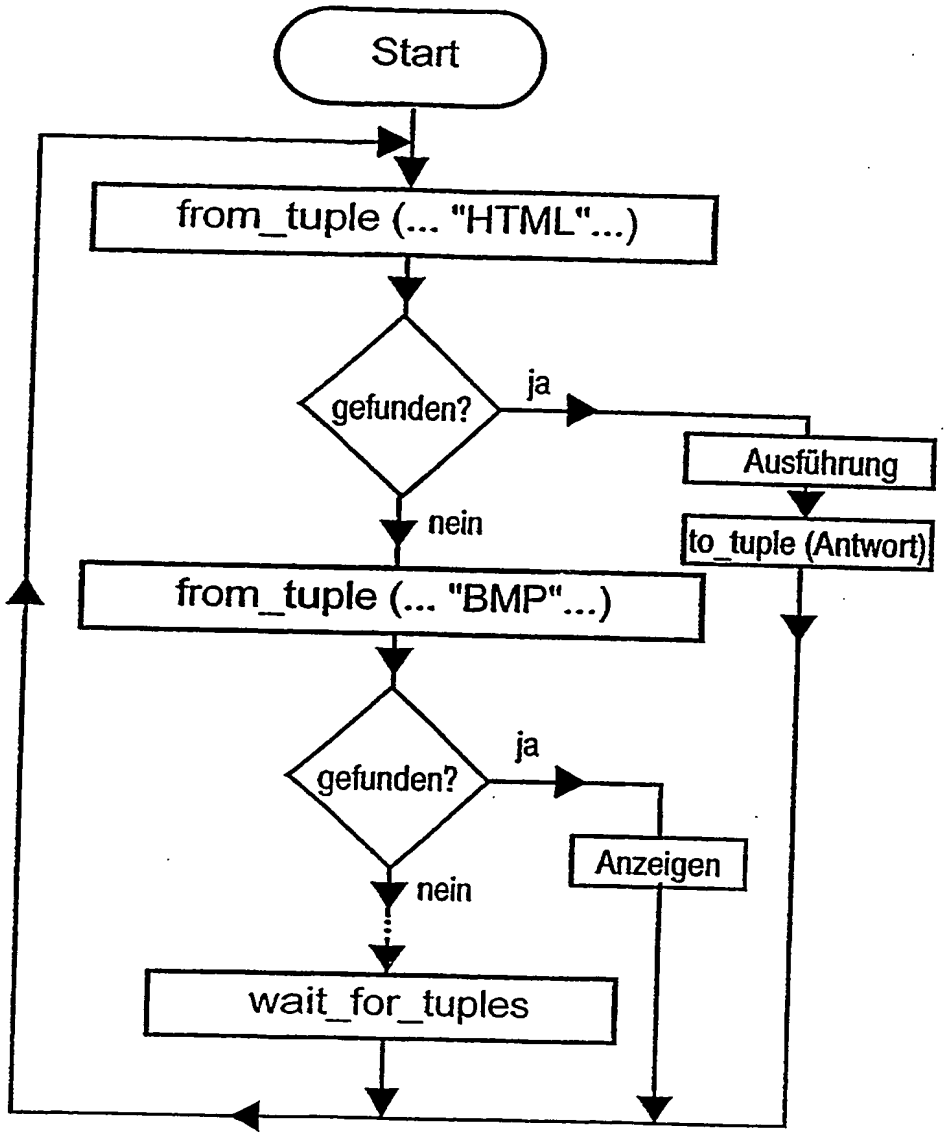

Fig. 3b

Num[redacted]
Int. Cl.:
Offenlegungstag:
DE 196 50 515 A1
H 04 N 7/025
25. Juni 1998

Umformer 124

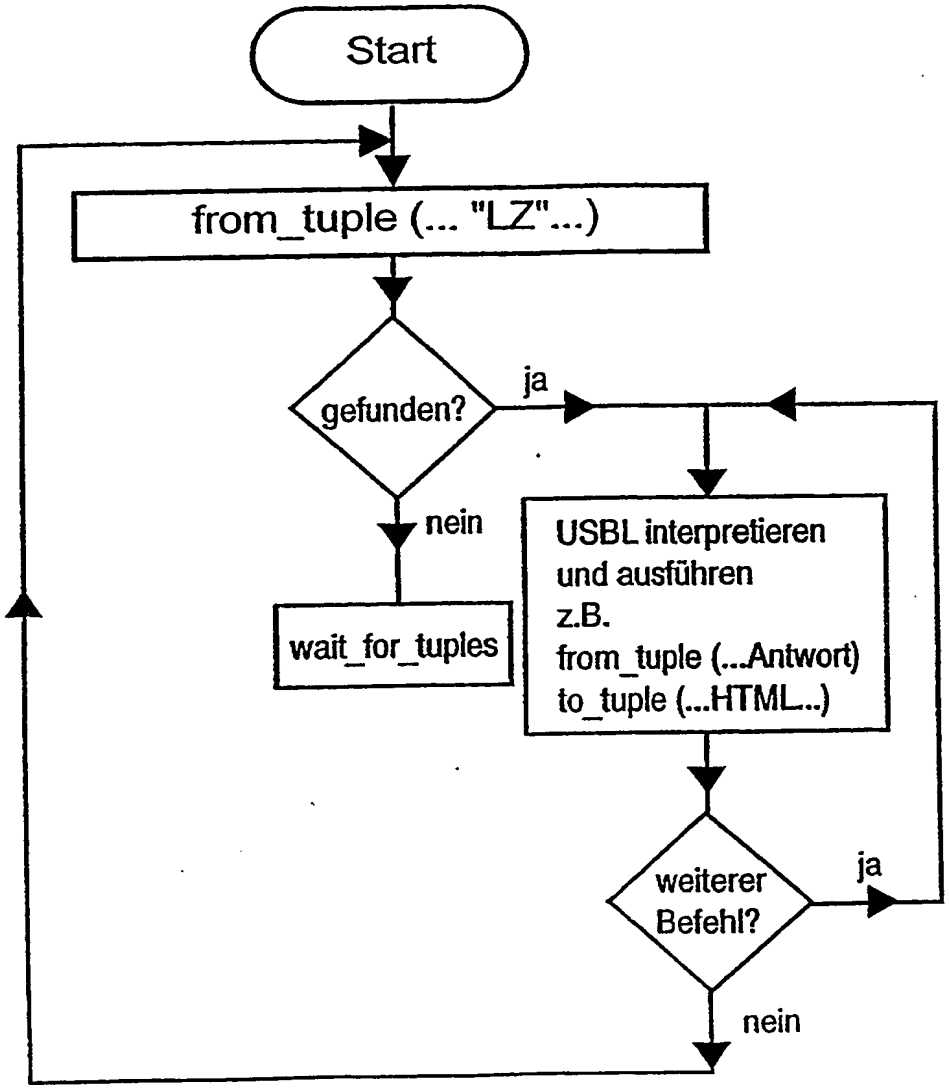

Fig. 3c

Num
Int. Cl.
Offenlegungstag:

DE 196 50 515 A1
H 04 N 7/025
25. Juni 1998

Systemsteuerung *122*

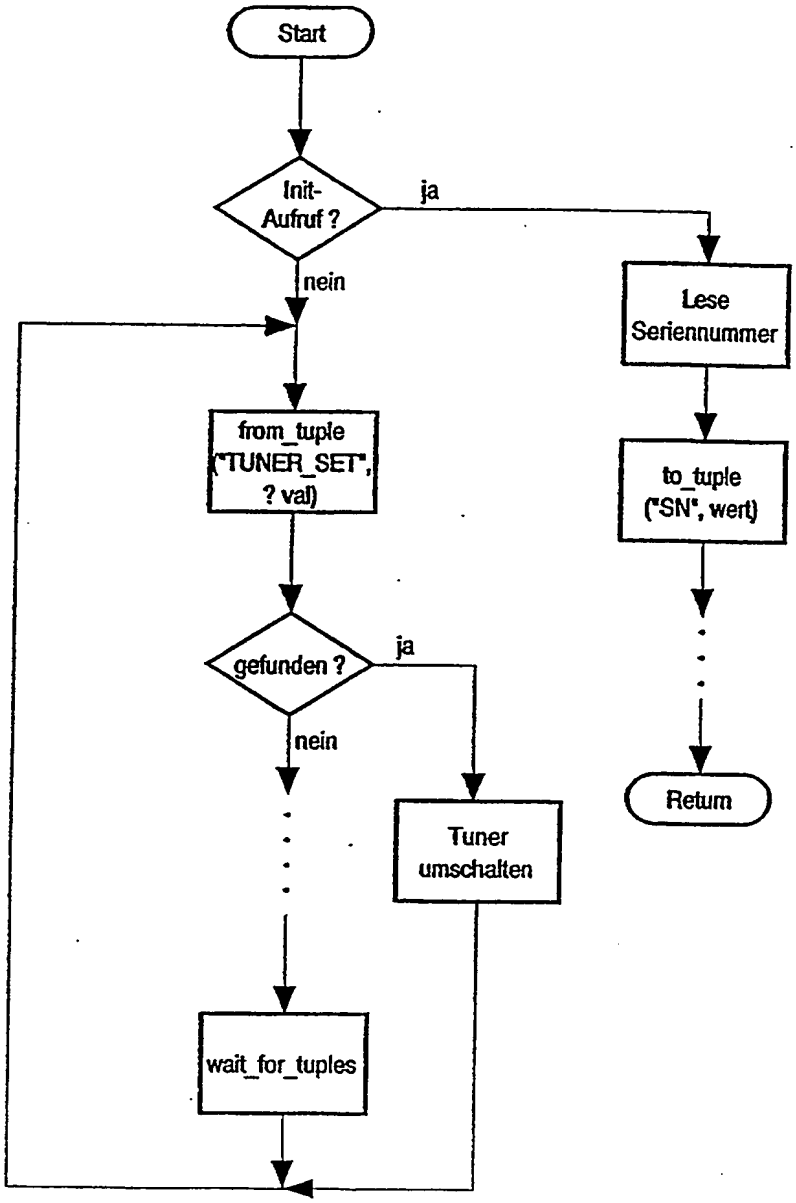

Fig. 3d

Num[redacted]:
Int. Cl.[redacted]:
Offenlegungstag:

DE 196 50 515 A1
H 04 N 7/025
25. Juni 1998

Empfangsroutine *1212*

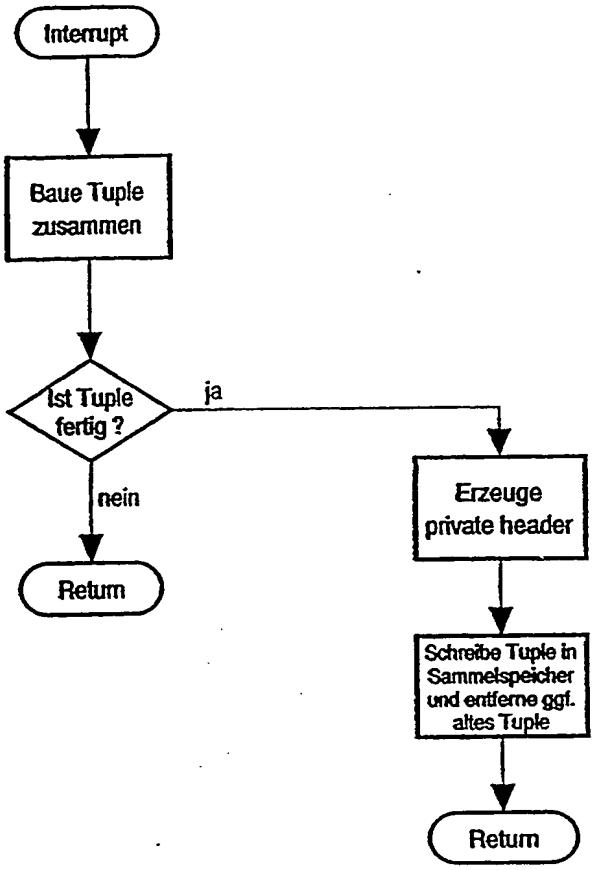

Fig. 3e

Num[illegible]:
Int. C[illegible]
Offenlegungstag:
DE 196 50 515 A1
H 04 N 7/025
25. Juni 1998

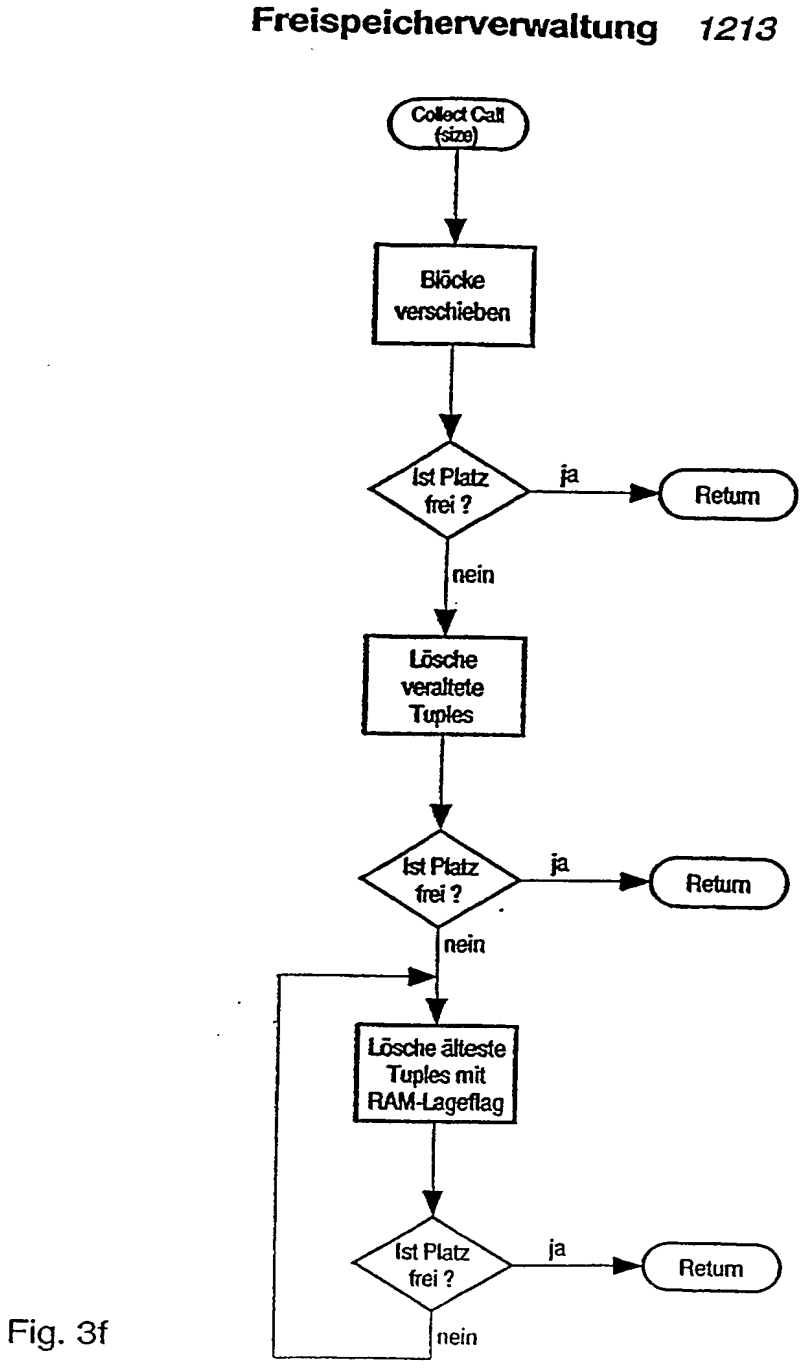

Fig. 3f

Nummer: DE 196 50 515 A1
Int. Cl.: H 04 N 7/025
Offenlegungstag: 25. Juni 1998

Betriebssystem *125*

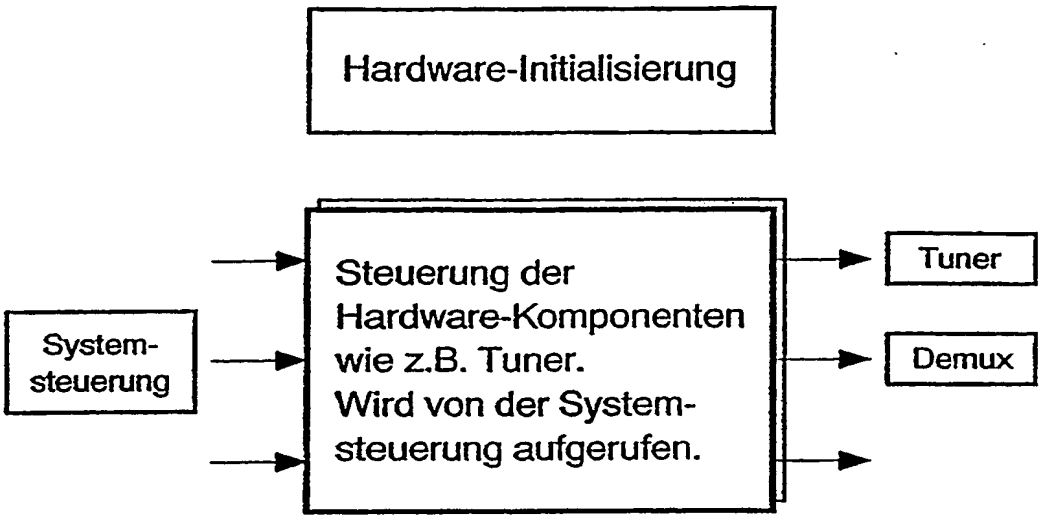

Fig. 3g

Num[redacted]
Int. Cl.:
Offenlegungstag:

DE 196 50 515 A1
H 04 N 7/025
25. Juni 1998

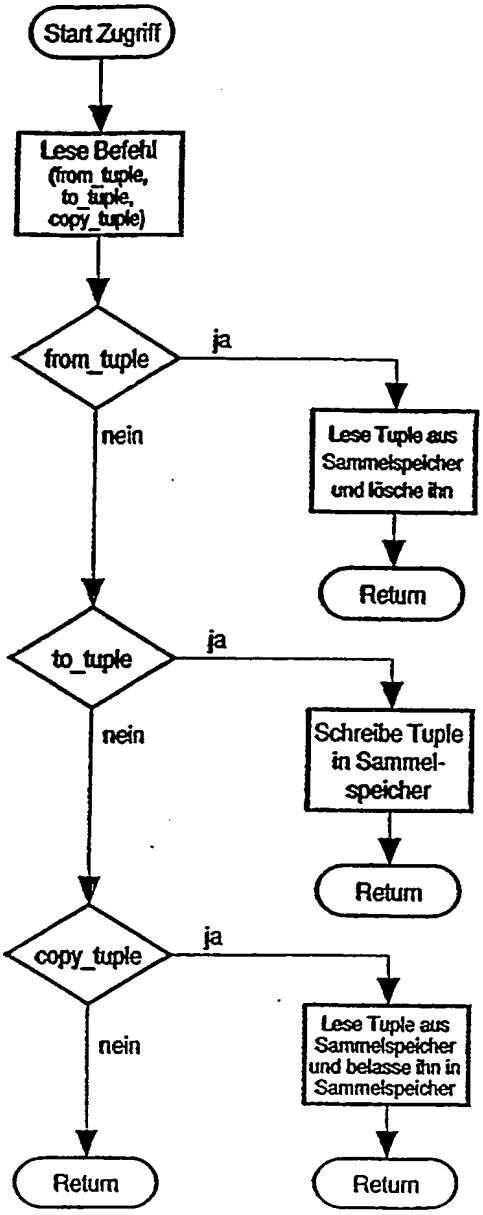

Fig. 3h

Numm
Int. Cl.
Offenlegungstag:
DE 196 50 515 A1
H 04 N 7/025
25. Juni 1998

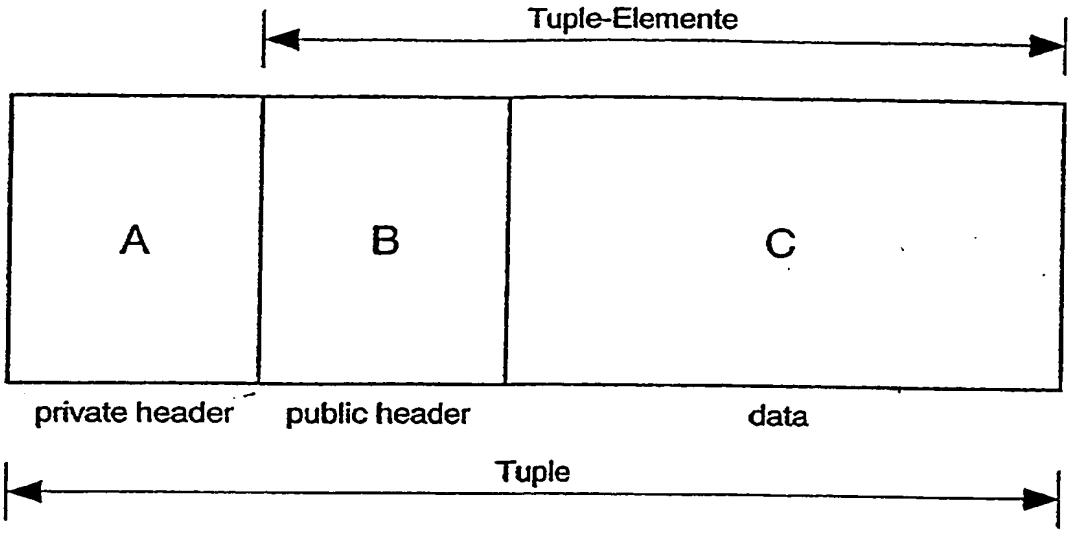

Fig. 4a Grundsätzliches Tuple-Format

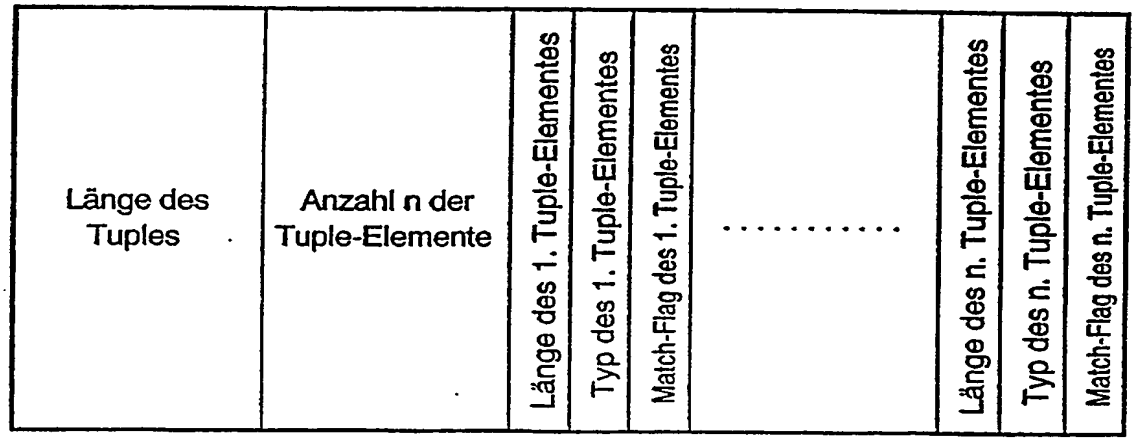

Fig. 4b private header

Nummer: DE 196 50 515 A1
Int. Cl.: H 04 N 7/025
Offenlegungstag: 25. Juni 1998

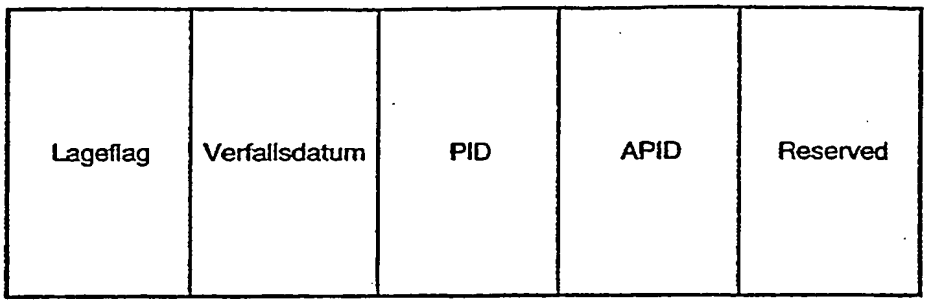

Fig. 4c public header

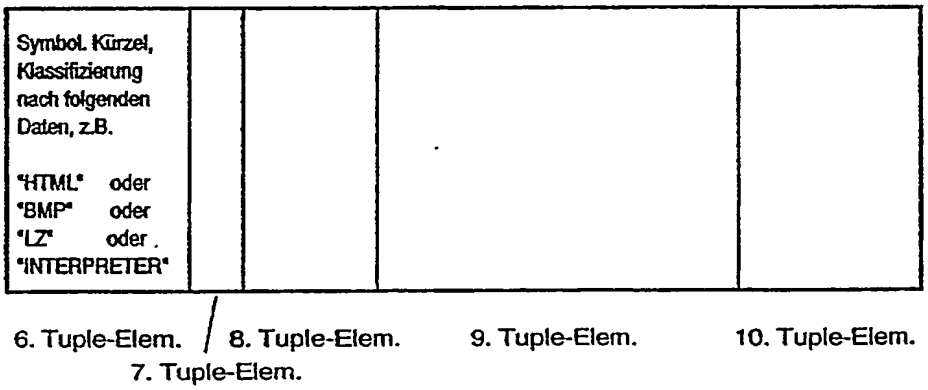

Fig. 4d Data